Dynamical Blueprints: Exploiting Levels of System-Environment Interaction

Nicolás S. Estévez
Computational Synthesis Lab
Dept. of Mechanical Engineering, Cornell University
Ithaca NY 14853, USA
+46 76 231 6994
nse5@cornell.edu

Hod Lipson
Computational Synthesis Lab
Dept. of Mechanical Engineering, Cornell University
Ithaca NY 14853, USA
(607) 255 1686
Hod.lipson@cornell.edu

ABSTRACT
Developmental systems typically produce a phenotype through a generative process whose outcome depends on feedback from the environment. In most artificial developmental systems, this feedback occurs in one way: The environment affects the development process, but the development process does not necessarily affect the environment. Here we explore a condition where both the developing system and the environment affect each other on a similar timescale, thus resulting in system-environment dynamical interaction. Using a model inspired by termite nest construction, we demonstrate how evolution can exploit these system-environment dynamics to generate adaptive and self-repairing structure more efficiently than a purely reactive developmental system. Finally, we offer a metric to quantify the level of interaction and distinguish between reactive and interactive developmental systems.

Categories and Subject Descriptors
I.2.2 [**Artificial Intelligence**]: Automatic Programming

General Terms
Algorithms, Design

Keywords
System-Environment Interaction, Developmental Systems, System Representation

1. INTRODUCTION
Phenotypic form of evolved systems can be represented through direct encoding or through a generative process. The term *developmental system* is often used to describe a generative process that produces a phenotype whose form depends on feedback obtained from the environment during development. This process is markedly different from *ballistic* generative systems which produce a final phenotype without such feedback. While purely ballistic processes are unlikely in any realistic environment, they have been successful in specifying structure in simulation [10][20]. The advantage of developmental systems, however, is their ability to adapt to the environment in situ [4].

In this paper we aim to define and investigate a fourth level of phenotypic representation which is based on bi-directional system-environment dynamics. In most artificial developmental systems (e.g. Bongard), the environment affects the development process, but the development process does not necessarily affect the environment (or affects it on a timescale that is relatively slow) [3]. We call this a *reactive* developmental system. In contrast, when both the developing system and the environment affect each other on a similar timescale, the resulting system-environment interaction leads to bi-directional dynamics that can be exploited by the evolutionary process. While most biological developmental systems may be interactive, most developmental systems studied in the evolutionary computation literature tend to be reactive. We believe that this distinction between reactive developmental systems and interactive developmental systems is critical to understanding many of the apparent complexities observed in biological systems, and can help generate more robust artificial systems as well.

1.1 Levels of system representation
When considering system-environment interaction one must be aware of the varying degrees of system-environment interaction and what those different levels mean. System-environment interaction is heavily studied even outside developmental systems. Environmental feedback and evolutionary algorithms can be combined to generate adaptive behavior in locomotive systems [1][3][15][18][19]. Even though none of these works are about developmental systems they explore system-environment interaction and also support the use of evolutionary algorithms linked with environmental feedback.

- **Explicit**

The representation with the least System-Environment Interaction is the **explicit level**. Systems at this level are represented as blueprints and the environment has no effect in the final form of the system because form is pre-determined within their representation.

- **Ballistic**

Next we have the **ballistic level**, at this level the system develops by following a sequence of instructions without using any kind of feedback, affecting the environment but with no knowledge of how the system is affected by the environment. An example of this level of system-environment interaction could be a system operating with an open-loop controller that takes no input and does not make any adjustments to it behavior. As generative systems they still offer advantages over conventional blueprints as a method of form specification since they have been shown capable of exploiting environment physics [20].

- **Developmental (reactive)**

At a **reactive level** we have systems that use environment feedback during their development but because of their timescale, environment dynamics do not affect development. A reactive system would use a closed-loop controller but would act fast so that its development would be mostly affected by the initial state of the environment. Even though this controller uses feedback during operation it is mostly just aware of the direct effects of its own actions and is not aware of changes in environment dynamics and much less able to react to them. When the system develops on a timescale that is much faster than the environment dynamics, the system is basically not aware of these dynamics. If the environment is not dynamic then the reactive level is the highest level of system environment interaction a system can achieve in that environment [2][5][9][12]. At this level of system-environment interaction the system is already able to adjust its developmental path to suit different environments.

- **Developmental (interactive)**

At the **interactive level** a system uses dynamic environmental feedback by developing on a timescale that allows the environment to react to its actions. Thus, this system is able to exploit the environment's dynamics [17]. During the system's developmental stage the system perturbs the environment through its actions and because of these perturbations the environment will be developing alongside the system using system feedback just like the system uses environmental feedback. Therefore, the environment dynamics can affect the behavior and development of the system. This parallel development allows the agent access to information about the environment that is simply unavailable to reactive systems. The interactive level is only possible to achieve when the timescale of the environment's dynamics is faster or comparable to the timescale of the system's development. Feedback Systems using the same channels of perturbation could potentially fall under the reactive or interactive levels of system-environment interaction based solely on a difference in timescales.

Previous work has been done comparing ballistic vs. reactive systems [10] although this terminology was not used. The experiments included in this work where designed to find what effect on performance occurs from using similar systems on different timescales, and this was done by a comparison of reactive vs. interactive systems. The experiments mentioned are all evolutionary runs using a hill climber evolutionary algorithm.

2. SYSTEM AND ENVIRONMENT

The environment and experiments were designed as a two-dimensional simplification of how termites build their nests and the environment in which this is done. The accuracy of the model relating to termites is not a priority but the terminology regarding the physics and constraints do revolve around this concept. Our system is situated in an environment consisting of a two-dimensional square grid composed of square cells. Each square cell has a floating-point temperature value which is represented in the simulation as a color (Fig.1). The temperatures are allowed to go beyond 0 and 100 degrees (of some arbitrary unit) however, the color of a cell will only show a gradient between 0 and 100 degrees. This constant color map was chosen to ease comparison between results.

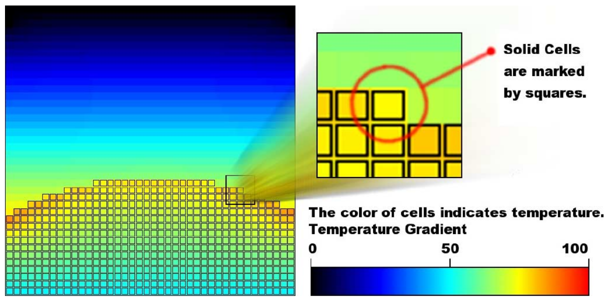
Solid Cells are marked by squares.
The color of cells indicates temperature.
Temperature Gradient
0
50
100

Figure 1 - Screenshot of environment showing the initial conditions of matter that is referred to as the "hill" and one of the initials conditions for temperature.

Aside from this floating-point temperature value, cells in these simulations also have a binary matter state. A cell can either be empty for matter state 0 or solid for matter state 1. In the display, solid cells have a black outline. The main difference between the matter states is that solid cells absorb sunrays (which results in an increase in temperature) while the empty cells will not. The other difference lies in their heat conduction and heat capacitance properties. In both cases the solid cells have a higher value. Figure 1 shows the starting conditions of solid and empty cells and one of the three starting conditions for temperature. A hill shape was chosen so that different initial conditions for temperature would result from each of the sunray angles.

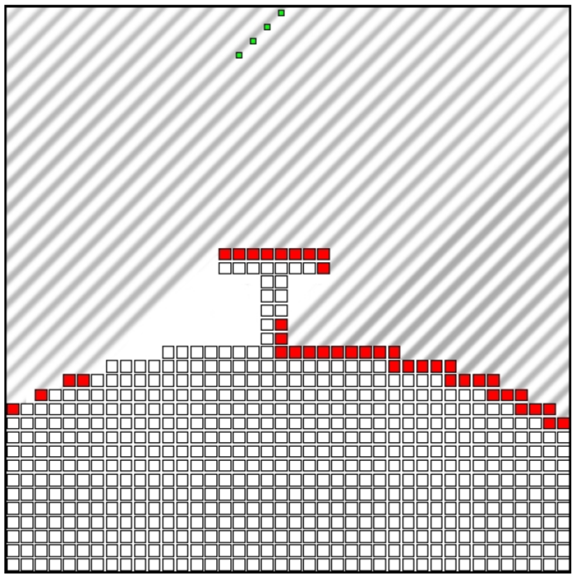

Figure 2 - Shading pattern that would result from a T-shaped structure and a sun angle of 45 degrees (as in test 3).

The "sunrays" mentioned earlier refer to a part of the simulation where vector rays originating for the top of the environment and descending in parallel at a specified angle until they collide with a solid cell (Figure 2). This causes the cell's temperature to rise by a fixed amount. There is a constant temperature boundary condition placed on the top and bottom rows of cells. These rows act as heat sinks to the solar input so that steady state is achieved within our defined temperature color scale. This defines the environment for the system. The system used for these experiments executes an action each time step. The action taken by the system follows the rule defined by its genome as shown in Figure 3.

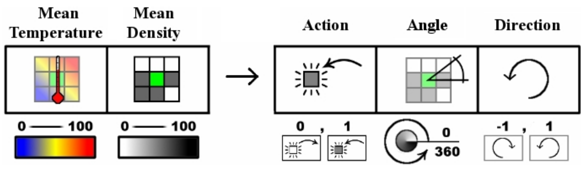

Figure 3 - Breakdown of system genome. First two genes represent the conditions and the last 3 genes specify the action.

In each time step the system scans only the empty cells that are adjacent to at least one solid cell to find a location in the environment that best matches its first two genes for mean temperature and mean density. Once a location is selected the action is taken according to the last 3 genes. The first action gene specifies whether the system builds or digs (action, 1 or 0). The next gene specifies the direction of the building or digging (angle, 0 to 360). The system is not allowed to build on a solid block or dig an empty spot (waste of a turn), so a built-in safety has the system rotate until it finds a suitable cell for its action (for example, an empty cell for building). The direction of this rotation is specified by the last gene (Direction, -1 or 1).

The system in these experiments, as envisioned originally, represented termites. The systems actions reflected termites working on the surface of the soil (hence the constraint of scanning empty cells adjacent to a solid one). This however, presents a problem of disembodiment since the termites are not physically represented in the simulation. Further thought lead to the conclusion that the system is not just the termites but the whole nest. The soil and termites together create a termite/nest system which is indeed what is developing here. Once this point is made, the concepts fit better together as the genotype which from a different point of view seems to dictate behavior and not development, does indeed dictate the development of the nest. In Emerson's introduction to his work he makes the observation that a termite nest is the physical representation of termite behavior [6]. Thus the genotype to phenotype connection of our system can be said to be inspired by the connection between termite behavior and the structure of the nest. That leads to a comparison between the way our system develops and interacts with our environment and the way a termite/nest system develops and interacts with our environment. And just as the environment can affect the development of a termite nest [8] our environment can also affect the development of our system.

Therefore, our system can also be seen as a growing or shrinking solid mass with reacts to the environment by expanding or receding in certain spots of its surface. Thus our system is indeed embodied and is interacting with the environment.

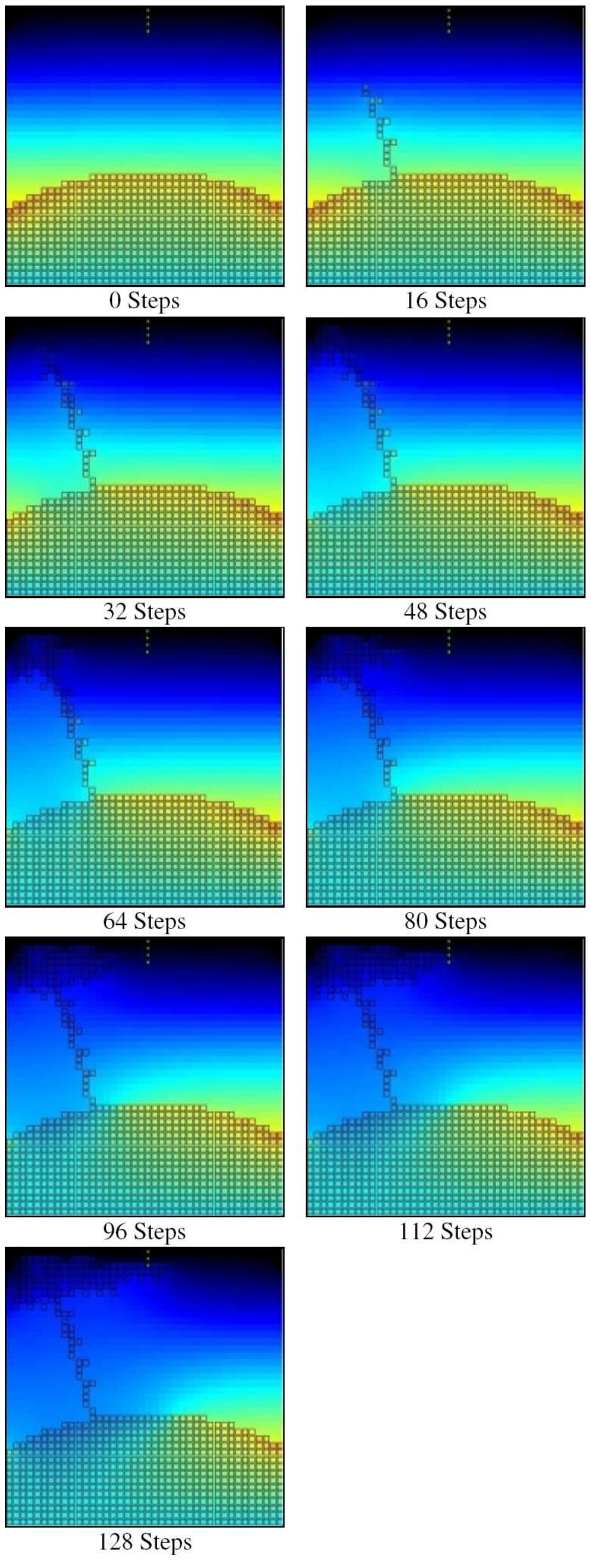

Figure 4 – Example for progression of test 1 on an interactive system. Notice the cell temperatures are being updated as the system builds.

3. EXPERIMENTS

The goal of our experiments was to find out how interactive systems would perform a given task in comparison to reactive systems. The task given to both systems was to have the temperature value of all cells be as close as possible to a certain value. As the system can only directly affect the matter states of the cells then the task is really to build a structure such that the dynamics of the environment cause all cells to be as close as possible to the target temperature. Fitness was calculated as one divided by the sum of differences between each cell's temperature and the target temperature. As a test for robustness, the final fitness appointed to a system was the average fitness of 3 different tests. The only difference between each test condition was the solar radiation angle. Figure 5 shows the corresponding solar angle for each test.

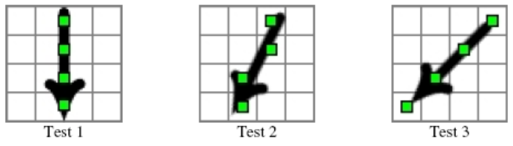

Figure 5 – Solar angle for each of the three tests and their green-square patterns as shown in the top of the display.

Each test begins with the environment in steady state according to a chosen tolerance as seen in the first frame of figure 4. In each test the systems are allowed 128 steps to create a structure. At the end of the test the system is allowed to reach steady state and then the fitness for that test is measured.

The difference between the reactive and interactive systems is that for the reactive system tests the environment does not update between steps while for the interactive systems the environment is allowed to reach steady state.

4. RESULTS

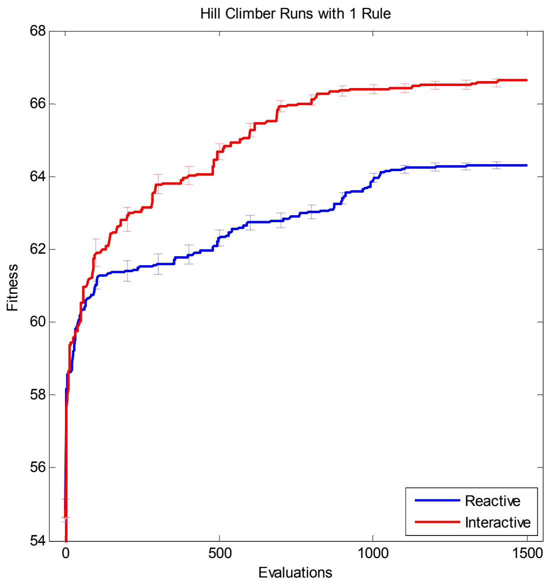

Figure 6 – Results averaged from 16 runs, error bars included. T-test significance = 0.0269

The main results of this work are shown in figure 6. Systems with an Interactive Level of system-environment interaction are indeed able to achieve a higher fitness performance when compared to reactive systems.

Taking a closer look at each of the 16 run's best individuals for each system gives further insight into what solutions the evolution was finding and what genotypes performed the best.

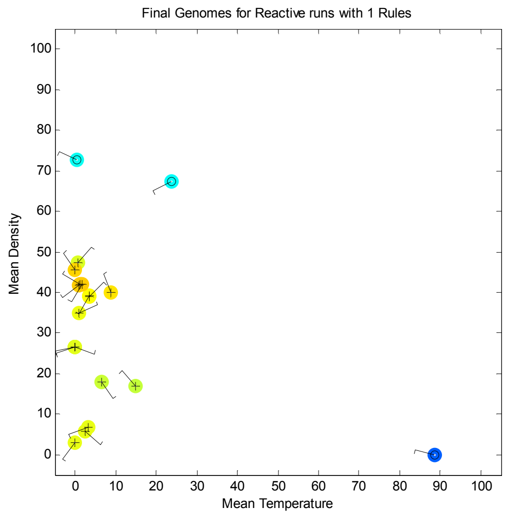

Figure 7 – Evolved Genotypes for Reactive Systems.

When reading Figures 7 and 8 note that each point or individual needs to express six dimensions of data (five genes plus the fitness value).

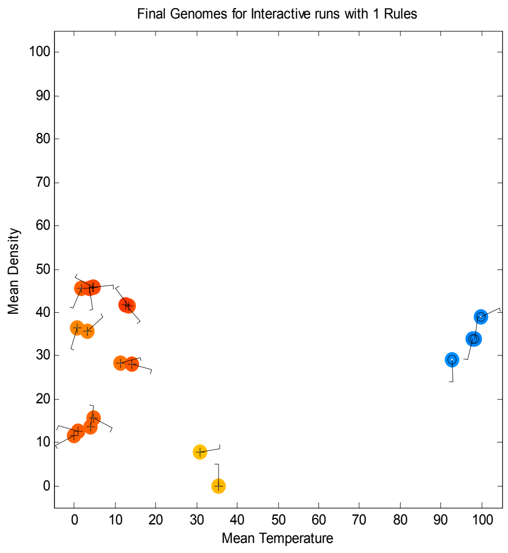

Figure 8 – Evolved Genotypes for Interactive Systems.

The first 2 dimensions are shown as the x and y coordinates on the graph. These correspond to the temperature and density sensor genes respectively. The next dimension shown corresponds to the binary action gene that indicates whether the system builds or digs. A cross symbol indicates a builder system and a circle symbol is an excavator. The angle and direction genes are displayed as the line extending out of the center towards the angle that the action was to be taken at the little knob at the tip showing whether it was a clockwise or counter-clockwise rotation. The final dimension is the fitness and it is shown by color. The same color gradient as the temperatures is shown with the highest fitness being red and the lowest being black.

In both figures 7 and 8, a strong correlation can be observed between fitness and the first three genes. Note how the best fitness values correspond to systems in the lower left corner of the plots and they are always builders. The digging systems where never able to evolve into high fitness and also do not show any preferred region in the sensor landscape. The angle and direction genes have no visible pattern. Therefore these genes must not have had a large effect on the structures being built. The main difference noted between the two graphs is that the interactive systems tended to have higher fitness (shown by the redness of their dots), so even though similar system genomes where evolved, in the case of the interactive systems better structures where being built.

Another question explored was regarding the performance of the systems in each individual test. In Figure 9 it can be seen that the performance in test one was always significantly lower than the performance in the two other tests. Was the system neglecting performance in test one to focus on tests two and three? To answer that question new experiments were run.

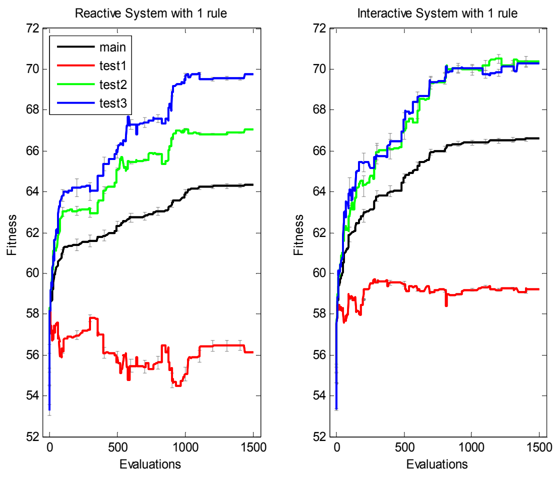

Figure 9 - Breakdown of results by test.

In order to explain these new experiments some new terminology needs to be introduced. These new experiments were evolutionary runs that focused on only one of the three tests at a time. So six new experiments where run, three tests for two kinds of systems. In the original experiments the systems where trying to optimize for all tests, while these new ones only focus on one test each. Thus, the new systems are called specialists and the originals are called generalists. Figure 10 shows an overlay of the originals systems over the specialists.

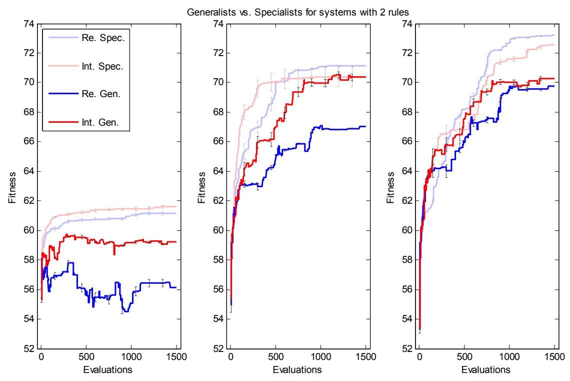

Figure 10 - Comparison of Generalists vs. Specialist. Showing tests 1, 2, and 3 from left to right.

The main take-away point from this graph is that even the specialist systems were not able to evolve good systems for test 1. Also note that for the specialist cases the reactive systems were better at evolving test 2 and 3, but for the generalist case the interactive system was able to make systems that where better in all three tests. The point here is that interactive systems have a performance increase when robustness is needed, but not necessarily when only a single problem/task needs to be solved. However, the motivation behind most research on systems powered by evolutionary algorithms is to seek automated adaptation in order to increase system robustness [11][23].

5. SYSTEM SELF-REPAIR

One of the main advantages of interactive developmental systems is that automated repair can be just part of the representation and execution of the system. In our case, the system simply determines where an action needs to be taken at the moment when it will be taken. If you remove some of the previously built structure the system will simply rebuild if its rules determine that this empty space is the best spot to build.

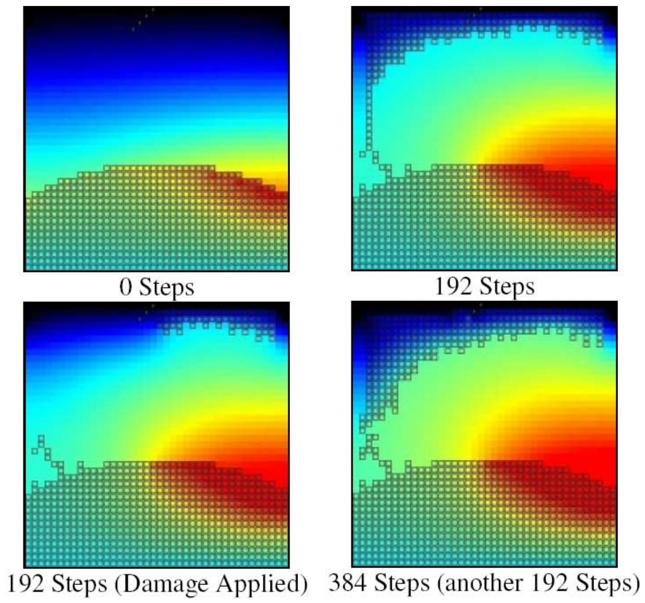

Figure 11 – Frames of test showing Self-repair

A designer would probably be tempted to include some sort of damage triggers and sequences that could be included among the rules of the systems to better allow a system to promptly fix damage, but in time with a better understanding of system-environment interaction we might end up realizing that it is not necessary to explicitly implement hard triggers and sequenced protocols, as these aspects of behavior will be implicitly evolvable. The following figure shows the self-repair capabilities of one of our evolved genomes.

Notice that the system rebuilds the part of the structure damaged at step 192. It does not rebuild exactly the same thing but the resulting structure serves the same purpose. This is reminiscent of regeneration in biological systems where a fully developed organism can replace lost parts [22]. Note that this system was not evolved for self-repair. The self-repair test was simply run on one of our previously evolved systems.

6. INTERACTIVE DYNAMICS INDEX

The relation of timescales between the system and the environment is what determines whether a system is reactive or interactive. Because there is a continuous scale from a fully reactive system to a fully interactive system a metric needs to be defined in order to categorize these systems as reactive or interactive.

A fully reactive system is a system that develops in a relatively static environment, meaning that the system development is instantaneous and from the systems' point of view the environment is indeed static. On the opposite end of the spectrum are fully interactive systems. These systems are constantly developing in an environment that is in steady-state. This means system development is so slow that after each system step the environment reaches steady-state before the system takes the sensor input for the next step.

An interactive dynamics index (IDI) is proposed that goes from zero to one. Fully reactive systems are considered completely non-interactive so they correspond to zero dynamic interactivity, while fully interactive systems would have an index value of one. In practical real-world systems it is impossible to devise a system with exactly zero or one dynamic interactivity due to the required speed of development being infinitely fast or slow, respectively. However, one must be able to determine if a given system is reactive or interactive. Where is the midpoint or system response time boundary where a system changes from being reactive to interactive?

By specifying an environment steady-state tolerance one is able to experimentally find a system response time where the system allows the environment to reach steady-state according to that tolerance, call this T_{ss}. Then, we consider a system as interactive if its response time is one-quarter or greater this value.

If this value is called T_b and then a plot of equation 1 is made, one will note that the system response time of T_b on the y-axis will correspond to an IDI value equal 0.5 on the x-axis.

$$-\log_2(1-t)*T_b \qquad \textit{(equation 1)}$$

Figure 12 plots response times on the y-axis from zero to infinity (fully reactive to fully interactive) against the Interactive Dynamic Index on the x-axis from 0 to 1 while at the same time having the boundary condition at 0.5 IDI (the midpoint).

The reason for finding T_{ss} first is that it is simple and feasible to find the response time for what would be an experimentally fully interactive system according to a specified tolerance. And the reason for choosing a factor of 4 between T_b and T_{ss}, besides being a nice low square number, is that when placing T_b at the midpoint of the IDI then T_{ss} will be just over 90% fully interactive (93.75%). This index was devised to facilitate future experiments concerning effects of different degrees of dynamic interactivity.

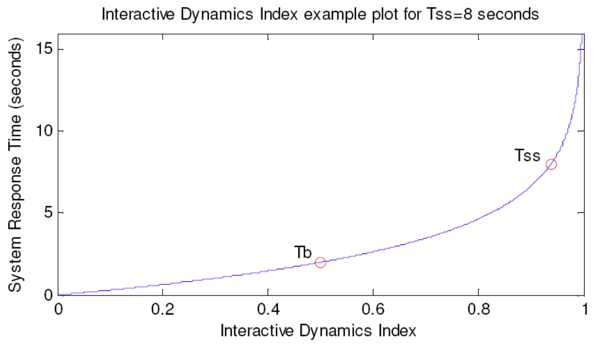

Figure 12 - Interactive Dynamics Index example plot.

7. CONCLUSION

Earlier we spoke about the different levels of system representation. Previous work by Rieffel and Pollack was mentioned that showed advantages of ballistic systems (term coined by Rieffel) over explicit systems (represented by blueprints) [20]. There is also previous work such as Hornby's tables that shows developmental systems (reactive) being advantageous when compared with ballistic systems [10]. Furthermore now a distinction has been made within developmental systems and two new levels of system representation have been created due to their different levels of system-environment interaction. Noting the correlation made several times in the past between increased system-environment interaction and performance, we set out to compare the Reactive and Interactive systems.

We have shown that systems evolved using an interactive level of system-environment interaction where able to build more functional structures over different environmental conditions. This shows more robust behavior. Increased adaptability is very important when designing systems that will need to build structures in unforeseen environments or if one knows that the system is going need to perform it's duties over a wide range of environmental conditions. When working with developmental systems it is important to pay attention to the level of system-environment interaction present, especially to identify the role that the environment plays in the development of the system. In biology the important role of the environment in the development of an organism is yet to become widely recognized [13].

The importance of looking closely at the system-environment interaction in systems that we work with goes beyond just achieving better performance and adaptability, it is also about having a better understanding of the behavior of our experimental system and seeing the importance of the role that the environment plays in system development.

8. FUTURE WORK

We will further explore the performance of reactive vs. interactive systems using different degrees of dynamic interactivity to find out if there is a noticeable threshold where the behavior changes from reactive to interactive. These experiments would result in an updated and validated Interactive Dynamics Index. Also, since the simple setup used for these experiments is able to show self-repair even when it wasn't even evolved for it, more work exploring the self-repair capabilities of an interactive system must be explored.

There is also potential in combining these self-repair capabilities with 3D printing technologies. Work is already being done in printing functional components with solid freeform fabrication using computer 3D models as blueprints [16]. With the addition of mobility this technology could eventually become the platform for robots that build the first structures using Functional Blueprints [7]. Already real-life reactive developmental systems have been built [21] so interactive systems is the next logical step.It would be of great interest to design interactive developmental systems tasked with solving real-world engineering problems starting in areas where evolutionary algorithms are being used for design already. There is great potential in the work done involving designing real-world systems [3][14].

Acknowledgements: This research was supported in part by the U.S. National Science Foundation grant number CMMI-0547376.

9. REFERENCES

[1] Almássy, Nikolaus P.W., Erik Vinkhuyzen. Evolution of Adaptive Behavior in Dynamic Environments. *In Intelligent Automation and Soft Computing: Trends in Research, Development and Applications*. Ed. M. Jamashidi, C.C. Nguyen, R. Lumia, and J. Yuh. Albuquerque, NM: TSI Press, 1994.

[2] Bentley, Katie, and Chris Clack. Morphological Plasticity: Environmentally Driven Morphogenesis. *ECAL* (2005): 118-127.

[3] Bongard, Joshua, Viktor Zykov, Hod Lipson. Resilient machines through continuous self-modeling, *Science*, (2006) Vol. 314. no. 5802, pp. 1118-1121.

[4] Bongard, Joshua, and Hod Lipson. Automated robot function recovery after unanticipated failure or environmental change using a minimum of hardware trials. *Proceedings NASA/DoD Conference on Evolvable Hardware* (2004): 169-176.

[5] Eggenberger, Peter. Evolving Morphologies of Simulated 3d Organisms Based on Differential Gene Expression. *4th European Conference on Artificial Life* (1997): 205-213.

[6] Emerson, Alfred E.. Termite Nests–A Study of the Phylogeny of Behavior. *Ecological Monographs* 8(1936): 247-284.

[7] Estévez, Nicolás. Funtional Blueprints: A Dynamical Approach to Structure Representation. *Master Thesis Cornell University* (2007)

[8] Harris, W. Victor. *Termites; Their Recognition and Control*. London: Longmans, 1961, 35.

[9] Hemberg, Martin, Una-May O'Reilly. Extending Grammatical Evolution to Evolve Digital Surfaces with Genr8. *EuroGP* (2004): 299-308.

[10] Hornby, Gregory S.. Functional Scalability through Generative Representations: the Evolution of Table Designs. *Environment and Planning B* 31.4(2004): 569-587.

[11] Hornby, Gregory S., Hod Lipson, and Jordan B. Pollack. Evolution of Generative Design Systems for Modular Physical Robots. *Proceedings IEEE International Conference on Robotics and Automation* 4(2001): 4146-4151.

[12] Kim, Jan T.. LindEvol: Artificial Models for Natural Plant Evolution. *KI* 14.1(2000): 26-32

[13] Lewontin, Richard. *The Triple Helix*. Cambridge, Massachusetts: Harvard University Press, 2000.

[14] Lipson, Hod, and Jordan B. Pollack. Automatic Design and Manufacture of Artificial Lifeforms. *Nature* 406(2000): 974-978.

[15] Mahdavi, Siavash H., and Peter J. Bentley. An Evolutionary Approach to Damage Recovery of Robot Motion with Muscles. *In 7th European Conference on Artificial Life* (2003): 248-255.

[16] Malone, Evan. Functional Freeform Fabrication for Physical Artificial Life. *Proceedings ALIFE 9* (2004): 100-105.

[17] Nagpal, Radhika. Programmable Self-Assembly using Biologically-Inspired Multiagent Control. *AAMAS* (2002).

[18] Pfeifer, Rolf, and Fumiya Iida. Morphological computation: Connecting body,brain and environment. *Japanese Scientific Monthly* 58.2(2005): 48–54.

[19] Quick, Tom, Chrystopher L. Nehaniv, Kerstin Dautenhahn, and Graham Roberts. Evolving embodied genetic regulatory network-driven control systems. *ECAL*. (2003)

[20] Rieffel, John, and Jordan B. Pollack.Automated Assembly as Situated Development: Using Artificial Ontogenies to Evolve Buildable 3D Objects. *GECCO* (2005): 99-106.

[21] Werfel, Justin, Yaneer Bar-Yamyz, Daniela Rus, and Radhika Nagpalz. Distributed Construction by Mobile Robots with Enhanced Building Blocks. *Proceedings IEEE International Conference on Robotics and Automation*. (2006): 2787-2794.

[22] Wolpert, Lewis, Rosa Beddington, Thomas Jessell, Peter Lawrence, Elliot Meyerowitz, and Jim Smith. *Principles of Development*. 2nd. New York: Oxford University Press, 2002.

[23] Zykov, Victor, Joshua Bongard, Hod Lipson. Evolving Dynamic Gaits on a Physical Robot *Late Breaking Paper, Proceedings GECCO* (2004)